Creating Enterprise Business Applications Using Eclipse RCP

Eclipse Live Webinar 9/4/2008
Kevin P. Taylor
Introduction

• President and Founder, Obtiva
• Former editor: java.about.com
• Author of multiple training courses and published articles
• Presenter: EclipseCon, EclipseWorld, Agile Alliance Conference
• Technical lead on Eclipse RCP projects
Introduction

• Enterprise Applications
  – Software used to drive business or organizational goals
  – Examples
    • Accounting
    • Sales Force Automation
    • Manufacturing
    • eCommerce
Introduction

• Rich Desktop Applications
  – Candidate Technologies
    • Swing
    • SWT
    • .NET
    • Flex
  – Good choice if:
    • More Intense User Interaction Requirements
    • Hours Not Minutes Per Day
    • Unique Needs
Introduction

• Project Background
  – Tasked with Writing A Large, CRM-Style Data Entry Application
    • Considered Flex
    • Considered Swing
    • RCP Was Attractive
      – Native UI
      – Pre-Built RCP Infrastructure
        » Windowing
        » Update Manager
        » Views/Editors
Introduction

Eclipse RCP Did Not Provide Everything Out Of The Box
Agenda

• Dividing JEE and Rich Client Responsibilities
• Desktop Patterns and RCP
• JFace Data Binding
• Form Validation
• Eclipse Update Site Pros and Cons
• Client Side Database Strategies
Dividing JEE and Rich Client Responsibilities
Dividing JEE and Rich Client Responsibilities

• Issue: Business Logic
• Web-based clients correctly pushed logic into middleware
• How does a disconnected client access that logic?
• There had to be sharing of logic across tiers
Dividing JEE and Rich Client Responsibilities

• Issue: Authentication
  – Client authenticates user
  – Server authenticates client
Dividing JEE and Rich Client Responsibilities

• Issue: Validation
  – User input validation
  – Client-to-server data validation
Dividing JEE and Rich Client Responsibilities

• Issue: Data Synchronization
  – Data resides in both local and remote data stores
  – Server: Oracle, flat-files, Web services
  – Client: HSQLDB/Apache Derby
Dividing JEE and Rich Client Responsibilities

• Issue: Dirty Data
  – When data changes on the server, the client must be notified that a synchronization is required
  – When data changes on the client, the client can connect and initiate a synchronization
Dividing JEE and Rich Client Responsibilities

• Issue: Client Disconnected State
  – Local data store required
  – Degrade gracefully
  – Limit number of functions that require remote access
Dividing JEE and Rich Client Responsibilities

• Issue: Synchronizing Application Logic
  – Shared libraries (Java on server and client!)
  – Eclipse Update Manager
Agenda

- Dividing JEE and Rich Client Responsibilities
- **Desktop Patterns and RCP**
- JFace Data Binding
- Form Validation
- Eclipse Update Site Pros and Cons
- Client Side Database Strategies
Desktop Patterns and RCP

- Initially, Use Cases Were Simple
- Few Widgets Per Screen
- Basic Models Were Created
- So, Data Was Bound Via Simple Widget Listeners
Desktop Patterns and RCP

- Presentation Logic Became Complicated
- Numerous Properties to Synchronize
- Testing of Cross Wired Listeners and Models was Difficult to Impossible
- A Better Overall Strategy Was Needed
Desktop Patterns and RCP - MVC

© 2007, 2008 by Scott Delap, Kevin P. Taylor; made available under the EPL v1.0
Desktop Patterns and RCP - MVP

© 2007, 2008 by Scott Delap, Kevin P. Taylor; made available under the EPL v1.0
Agenda

• Dividing JEE and Rich Client Responsibilities
• Desktop Patterns and RCP
• JFace Data Binding
• Form Validation
• Customizing the Eclipse RCP Look and Feel
• Eclipse Update Site Pros and Cons
• Client Side Database Strategies
JFace Data Binding

• Issue: How to Synchronize Changes
• Solution: MVP and JFace Data Binding
JFace Data Binding

• Robust, Flexible Design
• Lifecycle Hooks Useful for Other Application Events
  – Validation
  – Persistence
• Custom Validators and Converters Created
• Easy to Support New Class Types
Agenda

• Dividing JEE and Rich Client Responsibilities
• Desktop Patterns and RCP
• JFace Data Binding
• Form Validation
• Customizing the Eclipse RCP Look and Feel
• Eclipse Update Site Pros and Cons
• Client Side Database Strategies
Form Validation

• Issues:
  – Validation of Object Graph for Errors
  – Presentation of Those Errors In UI
Form Validation

• Validation of Object Graphs
  – Initial Implementation Used Apache Commons Validation
  – More Suited for Struts Style Form Beans Not Graphs of Objects
  – Callbacks Clunky
  – Recursion Difficult
Form Validation

- Validation of Object Graphs
  - Custom Approach was Required
  - Validation Registry of Validator Per Class
  - Each Validator Is Responsible For:
    - Validations of Object It Is Applied To
    - Recursing and Invoking Validation for Child Objects
Form Validation

• Created framework to tie presentation components to validation state of data.
• Each component’s appearance changed based on whether it’s data was valid or not valid.
  – E.g. red label, error icon, etc
Agenda

• Dividing JEE and Rich Client Responsibilities
• Desktop Patterns and RCP
• JFace Data Binding
• Form Validation
• **Eclipse Update Site Pros and Cons**
• Client Side Database Strategies
Eclipse Update Site Pros and Cons

• Issue: Updating Application Logic
  – Eclipse Update Manager provides several options for keeping client logic up to date
    • Manual updates
    • Programmatic updates
Eclipse Update Site Site Pros and Cons

• Issue: Update Manager Slow
• Needed to Optimize Updates
  – Keep plug-ins:
    • Small
    • Cohesive
  – 3rd party libraries get their own plug-in
Eclipse Update Site Pros and Cons

• Issue: Update Manager Still Slow
• Wrote a Custom Update Manager
  – Sent differences to client instead of complete plug-ins
Agenda

• Dividing JEE and Rich Client Responsibilities
• Desktop Patterns and RCP
• JFace Data Binding
• Form Validation
• Customizing the Eclipse RCP Look and Feel
• Eclipse Update Site Pros and Cons
• Client Side Database Strategies
Client Side Database Strategies

- Issue: Disconnected state == local data store required
- Candidate Technologies
  - MySQL (OS-specific)
  - Oracle Database Lite (expensive)
  - Prevayler (no query engine)
  - XML (no query engine)
  - HSQLDB
Client Side Database Strategies

• We initially chose HSQLDB
  • 100% pure Java
  • Allows a single directory install (no registry or OS ties)
  • Reasonable SQL support
  • Small footprint
  • Free

• We ended up with Apache Derby
Client Side Database Strategies

• Issue: Object-to-Relational Impedance

• We chose Hibernate
  – Object to Relational Mapping
  – Increased short-term productivity
  – Initially used long-running session
  – Eventually migrated to short-lived session
Client Side Database Strategies

• Issue: Hibernate introduced
  – Slow startup time
  – Relationship management complexity

• Slow startup time forced another solution
Client Side Database Strategies

• iBATIS Data Mapper
  – Simpler than Hibernate (less ambitious)
  – Allows developers to leverage existing SQL skills
  – Provides simple mapping between SQL and stored procedures
Client Side Database Strategies

• Issue: Report Generation
  – BIRT
    • At the start of our project, not considered mature
  – We chose JasperReports
    • Mature
    • Allows reporting against Java objects
    • We found iReport unstable and buggy
Conclusion

• Eclipse RCP Provides A Good Foundation
• There is Still Work Left To Do However
• API’s Such As JFace Data Binding Can Assist In These Tasks
• Design plug-ins with size and cohesiveness in mind
• Considerations Must Be Given To Concepts Like Persistence
  – What is Good for the Server Might Not Be for the Client (Hibernate)
• Disconnected state == local data store
Resources

- JFace Databinding
- HSQLDB
  - http://hsqldb.org/
- Hibernate
  - http://www.hibernate.org
- iBATIS
  - http://ibatis.apache.org/
- “Customizing Eclipse RCP applications,” IBM developerWorks
Resources

• “An Introduction to Data Binding,” IBM developerWorks